AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method for generating code to perform anticipatory prefetching for data references, comprising:

receiving code to be executed on a computer system;

analyzing the code to identify data references to be prefetched, wherein analyzing the code involves,

performing a first marking phase in which only data references located in blocks that are certain to execute are considered in determining which data references are covered by preceding data references, and

performing a second marking phase in which data references that are located in blocks that are not certain to execute are considered; and

inserting prefetch instructions into the code in advance of the identified data references, wherein inserting prefetch instructions includes inserting multiple redundant prefetch instructions for a given data reference;

wherein inserting multiple redundant prefetch instructions involves inserting the multiple redundant prefetch instructions into unused instruction slots, and wherein executing multiple redundant prefetch instructions potentially avoids a cache miss.

2. (Original) The method of claim 1, further comprising:

profiling execution of the code to produce profiling results; and

using the profiling results to determine whether a given block of instructions is executed frequently enough to perform the second marking phase on the given block of instructions.

3. (Original) The method of claim 2, wherein determining whether the given block of instructions is executed frequently enough to perform the second marking phase involves comparing a frequency of execution for the given block from the profiling results with a threshold value indicating a minimum frequency of execution to be considered in the second marking phase.

4. (Original) The method of claim 1, wherein analyzing the code involves:

identifying loop bodies within the code; and

identifying data references to be prefetched from within the loop bodies.

5. (Original) The method of claim 4, wherein if there exists a nested loop within the code, analyzing the code involves:

examining an innermost loop in the nested loop; and

examining a loop outside the innermost loop if the innermost loop is smaller than a minimum size or is executed fewer than a minimum number of iterations.

6. (Original) The method of claim 4, wherein analyzing the code to identify data references to be prefetched involves examining a pattern of data references over multiple loop iterations.

7. (Original) The method of claim 1, wherein analyzing the code involves analyzing the code within a compiler.

8. (Currently amended) A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for generating code to perform anticipatory prefetching for data references, the method comprising:

receiving code to be executed on a computer system;

analyzing the code to identify data references to be prefetched, wherein analyzing the code involves,

performing a first marking phase in which only data references located in blocks that are certain to execute are considered in determining which data references are covered by preceding data references, and

performing a second marking phase in which data references that are located in blocks that are not certain to execute are considered; and

inserting prefetch instructions into the code in advance of the identified data references, wherein inserting prefetch instructions includes inserting multiple redundant prefetch instructions for a given data reference;

wherein inserting multiple redundant prefetch instructions involves inserting the multiple redundant prefetch instructions into unused instruction slots, and wherein executing multiple redundant prefetch instructions potentially avoids a cache miss.

9. (Original) The computer-readable storage medium of claim 8, wherein the method further comprises:

profiling execution of the code to produce profiling results; and

using the profiling results to determine whether a given block of instructions is executed frequently enough to perform the second marking phase on the given block of instructions.

10. (Original) The computer-readable storage medium of claim 9, wherein determining whether the given block of instructions is executed frequently enough to perform the second marking phase involves comparing a frequency of execution for the given block from the profiling results with a threshold value indicating a minimum frequency of execution to be considered in the second marking phase.

11. (Original) The computer-readable storage medium of claim 8, wherein analyzing the code involves:

identifying loop bodies within the code; and

dentifying data references to be prefetched from within the loop bodies.

12. (Original) The computer-readable storage medium of claim 11, wherein if there exists a nested loop within the code, analyzing the code involves:

examining an innermost loop in the nested loop; and

examining a loop outside the innermost loop if the innermost loop is smaller than a minimum size or is executed fewer than a minimum number of iterations.

13. (Original) The computer-readable storage medium of claim 11, wherein analyzing the code to identify data references to be prefetched involves examining a pattern of data references over multiple loop iterations.

14. (Original) The computer-readable storage medium of claim 11, wherein analyzing the code involves analyzing the code within a compiler.

15. (Currently amended) An apparatus that generates code to perform anticipatory prefetching for data references, comprising:

a receiving mechanism that is configured to receive code to be executed on a computer system;

an analysis mechanism that is configured to analyze the code to identify data references to be prefetched, wherein the analysis mechanism is configured to,

perform a first marking phase in which only data references located in blocks that are certain to execute are considered in determining which data references are covered by preceding data references, and to

perform a second marking phase in which data references that are located in blocks that are not certain to execute are considered; and

an insertion mechanism that is configured to insert prefetch instructions into the code in advance of the identified data references, wherein inserting prefetch instructions includes inserting multiple redundant prefetch instructions for a given data reference<u>;</u>

<u>wherein inserting multiple redundant prefetch instructions involves inserting the multiple redundant prefetch instructions into unused instruction slots, and wherein executing multiple redundant prefetch instructions potentially avoids a cache miss</u>.

16. (Original) The apparatus of claim 15, further comprising a profiling mechanism that is configured to profile execution of the code to produce profiling results;

wherein the analysis mechanism is configured to use the profiling results to determine whether a given block of instructions is executed frequently enough to perform the second marking phase on the given block of instructions.

17. (Original) The apparatus of claim 16, wherein the analysis mechanism is configured to compare a frequency of execution for the given block from the profiling results with a threshold value indicating a minimum frequency of execution to be considered in the second marking phase.

18. (Original) The apparatus of claim 15, wherein the analysis mechanism is configured to:

identify loop bodies within the code; and to

identify data references to be prefetched from within the loop bodies.

19. (Original) The apparatus of claim 18, wherein if there exists a nested loop within the code, the analysis mechanism is configured to:

examine an innermost loop in the nested loop; and to

examine a loop outside the innermost loop if the innermost loop is smaller than a minimum size or is executed fewer than a minimum number of iterations.

20. (Original) The apparatus of claim 18, wherein the analysis mechanism is configured to examine a pattern of data references over multiple loop iterations.

21. (Original) The apparatus of claim 15, wherein the apparatus resides within a compiler.

22-45 (Canceled).